Consistent Memory Management Conventions

Daniel Plakosh, Software Engineering Institute [vita[1]]

2005-09-27; Updated 2008-10-06 L4 / D/P, L[2]

The most effective way to prevent memory problems is to be disciplined in writing memory management code. The development team should adopt a standard approach and apply it consistently.

Development Context

Dynamic memory management

Technology Context

C, UNIX, Win32

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Standard C dynamic memory management functions such as malloc(), calloc(), realloc(), and free() [ISO/IEC 99] are prone to programmer mistakes that can lead to vulnerabilities resulting from buffer overflow in the heap, writing to already freed memory, and freeing the same memory multiple times (e.g., double-free vulnerabilities).

Description

The most effective way to prevent memory problems is to be disciplined in writing memory management code. The development team should adopt a standard approach and apply it consistently. Some good practices include the following:

- Use the same pattern for allocating and freeing memory. In C++, perform all memory allocation in constructors and all memory deallocation in destructors. In C, define create() and destroy() functions that perform an equivalent function.
- Allocate and free memory in the same module, at the same level of abstraction. Freeing memory in subroutines leads to confusion about if, when, and where memory is freed.
- Match allocations and deallocations. If there are multiple constructors, make sure the destructors can handle all possibilities.

Steadfast consistency is often the best way to avoid memory errors. MIT krb5 Security Advisory 2004-002[16] provides a good example of how inconsistent memory management practices can lead to software vulnerabilities.

In the MIT krb5 library, in all releases up to and including krb5-1.3.4, ASN.1 decoder functions and their callers do not use a consistent set of memory management conventions. The callers expect the decoders to allocate memory. The callers typically have error-handling code that frees memory allocated by the ASN.1 decoders if pointers to the allocated memory are non-null. Upon encountering error conditions, the ASN.1

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/268-BSI.html (Plakosh, Daniel)
16. http://web.mit.edu/kerberos/advisories/MITKRB5-SA-2004-002-dblfree.txt

decoders themselves free memory that they have allocated but do not null the corresponding pointers. When some library functions receive errors from the ASN.1 decoders, they attempt to pass the non-null pointer (which points to freed memory) to free(), causing a double-free.

This example also shows the value of setting dangling pointers to NULL.

References

[ISO/IEC 99] ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming languages — C*. International Organization for Standardization, 1999.

Pearson Education, Inc. Copyright